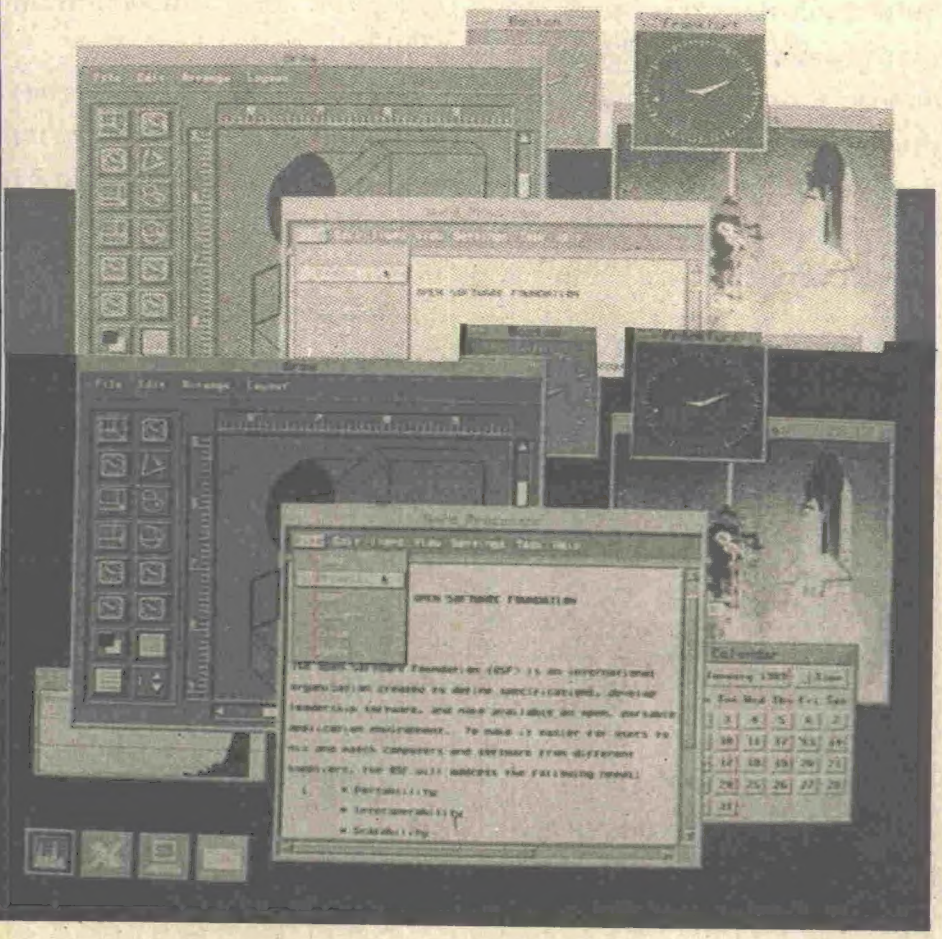
SOFT MAIL

# มาเขียนโปรแกรมใช้งาน ในระบบหน้าต่าง X กันเถอะ

*ศึกษาเบื้องหน้าเบื้องหลังการเขียนโปรแกรมภายใต้ X-WINDOW พร้อมทั้งมีโปรแกรมตัวอย่างประกอบในเรื่อง*



เมื่อไม่นานมานี้ โปรแกรมภาษาซีเริ่มระบาดไปทั่วโลก ตอนนั้นมีการโฆษณากันมโหฬารว่าภาษาซีดีอย่างนั้นอย่างนี้ มีโปรแกรมเขียนด้วยภาษาซีมาให้ใช้งานคือเวิร์ดสตาร์ ทำเอาเราเหล่าเซียนโปรแกรมสมองว่างเวลาน้อยมืออยู่ไม่สุขทั้งหลายเริ่มโรคจิตกำเริบอยากเขียนโปรแกรมภาษาซีกับเขาบ้าง จนกระทั่งมีท่านผู้รู้ทั้งหลายเริ่มเอาตัวอย่างภาษาซีมาให้ดู มีหนังสือมาให้อ่าน (คงรู้จักพระมหาคัมภีร์ซีของ K & R กันดี) มีคอมพายเลอร์มาให้แอบก๊อบปี้ใช้ อาการจึงได้ทุเลาลง แต่เราท่านรู้กันดีว่าโรคนี้ไม่มีทางหายขาด

มาถึงตอนนี้ระบบหน้าต่างหลายบานหรือพหุบัญชร X-Window เริ่มมาแรง มีการบรรยายสรรพคุณสารพัด อาการโรคจิตเก่าย่อมต้องกำเริบเป็นธรรมดา เรามาดูกันให้ถนัด ๆ เลยดีกว่าว่าจะเขียนโปรแกรมให้ใช้งานในระบบเอกซ์วินโดว์ต้องทำอย่างไรบ้าง จึงจะมีหน้าต่างของเราเอง ทำอะไรต่อมิอะไรได้อีกหลายอย่างด้วยโปรแกรมเราเอง เสียตั้งแต่ตอนนี้ เพราะถ้าขืนทิ้งไว้ก็มีแต่จะนั่งงง!

## หลักการเบื้องหลัง X

บรรดาเหล่านักโปรแกรมมือฉมังทั้งหลายก็ต้องพยายามเรียนรู้ที่จะใช้ภาษาหรือไลบรารี่ของโปรแกรมใหม่ ๆ ส่วนมากจะชอบที่จะได้ดูตัวอย่างโปรแกรมที่เขียนด้วยภาษา (หรือไลบรารี่)

นั้น ๆ เลยตั้งแต่ต้น เพื่อความคุ้นเคยและจะได้เริ่ม"คลำ"ตามไปได้ทันที เมื่อติดขัดอันใดแล้วค่อยกลับมาอ่านหนังสือคู่มือเป็นเรื่องเป็นราวเสียหนึ่งครั้งแล้วจึงกลับไป "คลำ" ต่อ วิธีนี้เหมาะสมสำหรับท่านนักโปรแกรมสมองใสที่ต้องการ"ออกตัวเร็ว" คือสามารถผลิตผลงานขนาดพอดูได้ในเวลาอันสั้นมากจากการอาศัยประสบการณ์จากภาษาเก่าผสมกับการตัดส่วนของตัวอย่างมาประรวมกันเข้า แต่เกือบทุกท่านจะพบว่าหลังจากทำอย่างนี้ไปได้สักพักหนึ่งจะเริ่มถึงทางตัน ไม่อาจจะใช้ขีดความสามารถของภาษาหรือไลบรารี่ใหม่ได้เต็มที่ เพราะในตัวอย่างจะอย่างไรก็บอกไม่หมด มีส่วนลึก ๆ ซึ่งบางครั้งมีประโยชน์แต่ละเอียดยิบเยื่อยากที่จะอธิบายไว้ในคู่มือได้หมด ในกรณีของ X นั้นแม้จะเป็นของใหม่แต่ความคิดพื้นฐานของการเขียนโปรแกรมนั้นเป็นของง่าย ๆ ค่อนข้างมาตรฐานที่ทุกคนคงทราบดีกันอยู่แล้ว ตั้งแต่ริอ่านจะเป็นนักเขียนโปรแกรมกับเขา ฉะนั้นถ้าจะสละเวลาสักนิดอ่านตรงส่วนนี้ก่อนก็จะเป็นการลงทุนครั้งแรกนี้ครั้งเดียวแล้ว "ไปโลด" กับการเขียนโปรแกรมใช้งานตามสไตล์ X

## Xlib คืออะไร

ถ้าจะเขียนโปรแกรมให้ใช้งานในระบบหน้าต่างเอกซ์วินโดแล้วละก็ หนีไม่พ้นต้องทำความรู้จักกับ Xlib เสียก่อน สำหรับผู้ที่ใช้งานภาษาซีมาแล้วก็คงจะพอเดาออกว่าคืออะไร Xlib คือ "รวมเล่ม"ของบรรดาโปรแกรมย่อยที่มีความจำเป็นต้องใช้ เพื่อให้โปรแกรมทำงานได้ตามความต้องการในระบบหน้าต่างเอกซ์วินโดว์ โดยที่เราไม่ต้องไปเขียนใหม่ และการใช้ Xlib ทำให้โปรแกรมของเรามีมาตรฐานสามารถนำไปใช้งานได้กับเครื่องทุกเครื่องที่มีเอกซ์วินโดว์ และ Xlib อยู่ นี่คือจุดเด่นด้านความสามารถในการย้ายโปรแกรมไปมาระหว่างเครื่องยี่ห้อต่าง ๆ โดยไม่ต้องเขียนโปรแกรมใหม่ แม้ว่าลักษณะเครื่อง เช่น ขนาดของจอภาพจะต่างกันก็ตาม

ข้อดีอีกอย่างหนึ่งของการใช้ Xlib ก็คือ Xlib นั้นถูกเขียนมาโดยมีระบบการโยงเครือข่าย (network) อยู่ในหัวใจ นักโปรแกรมสามารถจะให้เครื่องสองยี่ห้อคุยกันโดยผ่านระบบเครือข่ายได้อย่างง่ายดายโดยที่เมื่อก่อนมองดูแทบเป็นไปไม่ได้

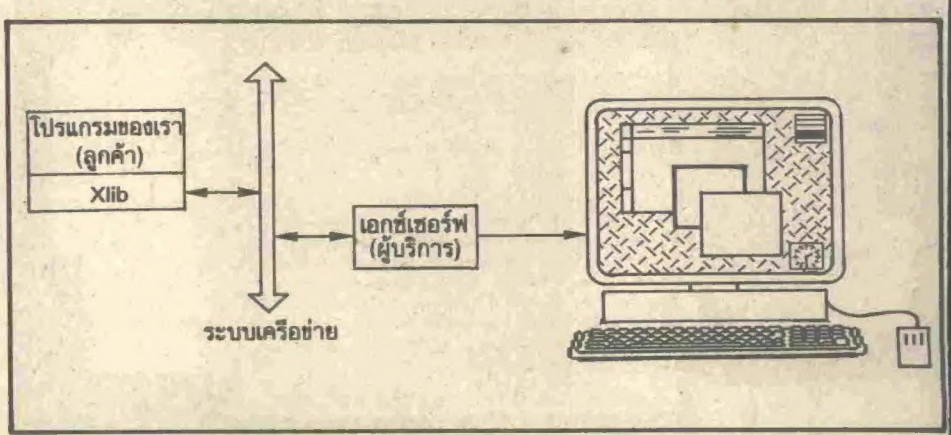
ข้อดีข้อสุดท้ายก็คือ ระบบเอกซ์วินโดว์นั้นเป็นระบบ "เปิด" นั่นคือไม่มีการกำหนดส่วนติดต่อกับผู้ใช้เป็นการเด็ดขาด ความ"เปิด"นี้จะมีประโยชน์เห็นได้ชัดก็คือเมื่อเครื่องคอมพิวเตอร์สองยี่ห้อมีความแตกต่างกันอย่างมากจนมาตรฐานที่กำหนดโดยอิงเครื่องหนึ่งเป็นหลักไม่สามารถทำงานได้กับเครื่องยี่ห้ออื่น ๆ

ถึงขณะนี้ แม้ว่าเราจำจะต้องเขียนภาษาซีและใช้ Xlib จึงจะทำให้โปรแกรมของเราใช้งานในระบบเอกซ์วินโดว์ได้ แต่ในอนาคตอันใกล้นี้ก็กำลังจะมีการพัฒนาให้สามารถเขียนด้วยภาษาอื่นได้ เช่น Lisp เป็นต้น

## การทำงานของ Xlib

เมื่อมีการเขียนโปรแกรมใช้งานในระบบเอกซ์วินโดว์ด้วย Xlib นั้น การทำงานของโปรแกรมจะมีการติดต่อระหว่างกันโดยใช้ "กรรมวิธีสื่อสาร" มาตรฐานอันหนึ่งเรียกว่า เอกซ์โปรโตคอล (X protocal) ซึ่งจะใช้ในติดต่อระหว่างโปรแกรมที่อยู่ต่างเครื่องกันผ่านระบบเครือข่าย รวมไปถึงกระทั่งโปรแกรมที่อยู่บนเครื่องเดียวกัน การทำเช่นนี้ทำให้นักโปรแกรมทั้งหลายไม่ต้องคอยห่วงว่าโปรแกรมแท้จริงแล้วอยู่ที่ไหน เพราะใช้มาตรฐานเดียวกันหมด ดังรูปที่ 1

เอกซ์เซอร์ฟเวอร์ (X server) จะเป็นผู้กุมทรัพยากรทั้งหมดของระบบ (จอภาพ, เมาส์, แป้นพิมพ์) ดังนั้น จึงทำหน้าที่เป็น "ผู้บริการ" ในขณะที่โปรแกรมที่เราเขียนขึ้นมานั้นเปรียบเสมือนผู้ไปขอใช้ทรัพยากรของเขา ดังนั้นโปรแกรมของเราจึงมีลักษณะเป็น "ลูกค้า" ไม่ว่าผู้บริการและลูกค้าจะอยู่




รูปที่ 1 ลักษณะการติดต่อสื่อสารในระบบ X-Window

บนเครื่องเดียวกันหรือไม่ก็ตาม หลักการอันนี้ก็ยังคงเดิมตลอดเวลา ฉะนั้นโปรแกรมของเราจะแสดงผลบนจอภาพได้ก็ดี จะรับคำตอบจากแป้นพิมพ์ได้ก็ดี จะตรวจการเลื่อนของเมาส์ได้ก็ดี จะต้อง "คุย" กับผู้ให้บริการทั้งสิ้น โดยใช้กรรมวิธีสื่อสารซึ่งมีแบ่งออกเป็น 4 ประเภท คือ

### 1. คำร้องขอ

เป็นการที่ Xlib ติดต่อกับผู้บริการเพื่อร้องขอให้ผู้บริการกระทำบางสิ่งบางอย่างให้ เช่น การลากเส้นบนจอภาพ เปลี่ยนสีจอภาพ หรือต้องการทราบข้อมูลเกี่ยวกับขนาดของหน้าต่าง(ซึ่งเปลี่ยนแปลงได้) อันที่จริงแล้วแทบทุกคำสั่งใน Xlib นั้นเป็นคำร้องขอทั้งสิ้น มีข้อยกเว้นอยู่ไม่กี่คำสั่ง ทั้งนี้เพราะผู้กุมอำนาจที่แท้จริงคือผู้บริการ จะทำอะไรก็แทบจะต้องพึ่งเขาไปหมด

### 2. คำตอบรับ

คือการที่ผู้บริการติดต่อกลับมายังโปรแกรมของเราผ่านทาง Xlib เนื่องจากการที่เราร้องขอบางสิ่งบางอย่างไป การร้องขอนั้นไม่จำเป็นต้องมีการตอบรับเสมอไป ปกติการตอบรับจะเกิดขึ้นเมื่อคำร้องขอนั้นเกี่ยวข้องกับความต้องการทราบข้อมูล ตามตัวอย่างที่เพิ่งผ่านมา การขอให้ลากเส้นบนจอภาพก็ดี การเปลี่ยนสีจอภาพก็ดี ไม่ต้องการคำตอบกลับมาจากผู้บริหารแต่อย่างใด ในทางตรงข้ามการขอทราบขนาดของหน้าต่างนั้นต้องการคำตอบกลับมาจากผู้บริการ ดังนั้นจะเห็นได้ว่าการร้องขอที่รอการตอบรับด้วยจะใช้เวลาและต้องติดต่อผ่านเครือข่ายเพิ่มขึ้นเป็นสองเท่า ซึ่งอาจทำให้โปรแกรมของเราทำงานช้าลงพอสมควรทีเดียว

### 3. การแจ้งเหตุ

การแจ้งเหตุเป็นการติดต่อสื่อสารอีกอันหนึ่งจากผู้บริการมายังโปรแกรมของเรา (ผ่าน Xlib ตามเคย)เกี่ยวกับการเปลี่ยนแปลงของทรัพยากรต่าง ๆ ที่ผู้บริการดูแลอยู่ทั้งทางตรง(ผู้ใช้เลื่อนเมาส์ ฯลฯ) และทางอ้อม(โปรแกรมอื่นเปิดหน้าต่างมาบังหน้าต่างของเรา) ซึ่งเป็นข้อมูลที่เป็นประโยชน์ต่อโปรแกรมของเรา ทางผู้บริการจะแจ้งเหตุเหล่านี้เข้ามาเองโดยไม่ต้องมีการร้องขอ และข้อมูลของเหตุเหล่านี้จะถูกเก็บเข้าคิวไว้ (ไม่ได้นำมาใช้งานทันทีเหมือนคำตอบรับ) ทำให้ต้องใช้แนวคิดในการจัดการกับเหตุการณ์ที่รับทราบมา ต่างจากการเขียนโปรแกรมธรรมดาอยู่บ้าง

### 4. การแจ้งข้อผิดพลาด

ที่เราชาวโปรแกรมรู้จักดีคือ error ซึ่งถ้ามองดูจริง ๆ แล้วก็เป็นการแจ้งเหตุประการหนึ่งเหมือนกัน แต่เป็นการแจ้ง "อุบัติเหตุ" ดังนั้นจึงต้องมีการจัดการเพิ่มเติมกว่าปกติ โดยที่ใน Xlib จะมีส่วนโปรแกรมอยู่สองท่อนสำหรับการแจ้งข้อผิดพลาดนี้ ท่อนหนึ่งใช้สำหรับ "กู้ภัย" อุบัติเหตุชนิดที่แก้ไขได้ ในขณะที่อีกท่อนหนึ่งใช้ "ประคองตัว" ในกรณีเกิดอุบัติเหตุชนิดแก้ไขไม่ได้ ส่วนของโปรแกรมทั้งสองส่วนนี้ผู้เขียนโปรแกรมจะเขียนของตัวเองมาใช้แทนก็ได้

## ระบบคำร้องใน Xlib

ปัญหาความไม่เคยชินอันหนึ่งจะเกิดขึ้นกับการเขียนโปรแกรมโดยใช้ Xlib ก็คือ การที่เราร้องขอบางสิ่งบางอย่างแล้วไม่ได้ตามที่ต้องการ เหตุเกิดขึ้นเพราะ Xlib นั้นจะ "ดอง" คำร้องขอของเราเอาไว้ แทนที่จะส่งคำร้องไปยังผู้บริการทันที ที่ทำเช่นนี้ด้วยความหวังดีว่า โปรแกร. เองเราจะได้ทำงานต่อไปโดยไม่ต้องรอติดต่อกับผู้บริการเพื่อยื่นคำร้องทุก ๆ ครั้ง เพราะคำร้องบางอย่างไม่ต้องการการตอบสนองอย่างเร่งด่วนจากผู้บริการ ยิ่งถ้าเป็นการติดต่อผ่านระบบเครือข่ายยิ่งเป็นการดีที่จะมัดรวมคำร้องหลาย ๆ อันเข้าด้วยกันแล้วส่งทีเดียวเพื่อป้องกันความล่าช้า แต่บางครั้งความหวังดีอันนี้ก็ส่งผลกระทบมาให้แก่เขียนโปรแกรมถึงกับหัวปั่นไปเหมือนกัน ตัวอย่างเช่น เราต้องการให้โปรแกรมส่งข้อความขึ้นบนจอภาพเพื่อถามผู้ใช้งานให้กรอกข้อความบางอย่างกลับมา ปกติก็จะทำโดยร้องขอให้ผู้บริการนำข้อความคำถามไปแสดงที่จอภาพ จากนั้นก็จะรอการแจ้งเหตุที่ผู้ใช้ตอบข้อความกลับมา มองดูก็ตรงไปตรงมาดีไม่น่ามีอะไรพลาด แต่ปรากฏว่า

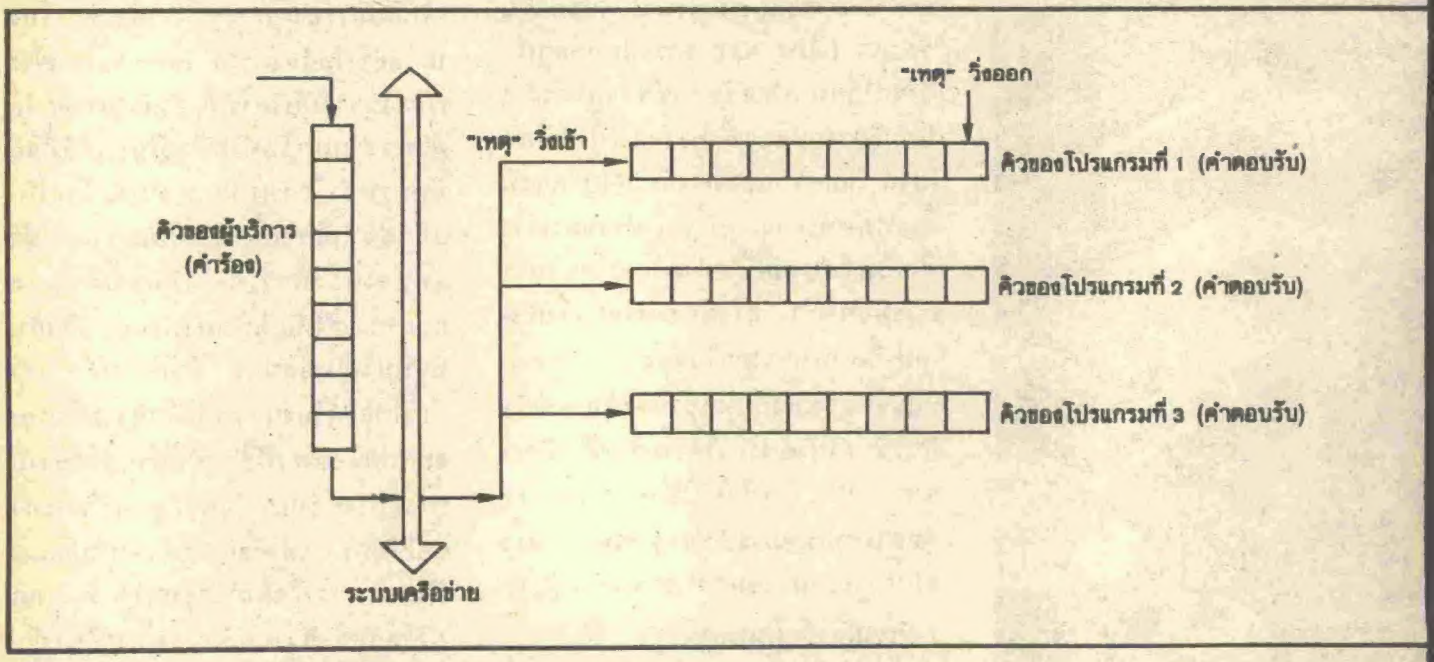
พอใช้งานโปรแกรมเข้าจริง ๆ มักจะมาหยุดอยู่ตรงนี้ เพราะผู้ใช้งานไม่เคยตอบข้อความกลับมา โถ! ก็จะตอบได้อย่างไร ตอนที่โปรแกรมเราร้องขอให้ผู้บริการแสดงคำถามบนจอภาพ Xlib จับคำร้องเราไปดองไว้ ก็เลยไม่มีคำถามปรากฏบนจอภาพ แล้วผู้ใช้งานคนไหนจะไปตรัสรู้ตอบคำถามที่ตัวเองมองไม่เห็นได้

การที่คำร้องทั้งหลายใน Xlib จะถูกส่งไปให้แก่ผู้บริการนั้นมี 3 กรณี หนึ่งคือ เมื่อโปรแกรมของเราต้องการรอการแจ้งเหตุเฉพาะอันใดอันหนึ่ง ในกรณีนี้โปรแกรมของเราว่างงานและจะต้องรอเหตุการณ์ให้เกิดขึ้นอยู่แล้ว Xlib จะจัดการส่งคำร้องต่าง ๆ ของเราไปพลาง ๆ เพื่อไม่ให้เรารอโดยเสียเวลาเปล่า กรณีที่สอง จะเกิดขึ้นเมื่อเราทำคำร้องพิเศษ(มักจะเป็นการร้องขอข้อมูลบางอย่าง) ซึ่งจำเป็นจะต้องได้รับการตอบจากผู้บริการทันที ในกรณีนี้เมื่อไหน ๆ จะต้อง "คุย" กับผู้ให้บริการอยู่แล้วก็ถือโอกาสส่งคำร้องทั้งหมดไปให้เสียเลยจะได้ติดต่อกันครั้งเดียวได้งานหลายอย่าง กรณีสุดท้ายคือ การใช้คำสั่งเฉพาะในการบังคับให้ Xlib จัดการยื่นคำร้องไปยังผู้บริการทันทีโดยไม่ต้องรอให้เกิดกรณีแรก ๆ เสียก่อน ซึ่งคำสั่งบังคับอันนี้ทำให้เราเหล่าผู้เขียนโปรแกรมสามารถบังคับให้ Xlib จัดการยื่นคำร้องขอตามจังหวะเวลาที่เราต้องการได้

นอกจาก Xlib จะดองคำร้องขอของโปรแกรมของเราแล้ว ยังมีการ"จัดคิว"คำตอบรับหรือการแจ้งเหตุจากผู้บริการมายังโปรแกรมของเราด้วยเช่นกัน ความรู้อันนี้เป็นสิ่งจำเป็นสำหรับบรรดาเขียนโปรแกรมทั้งหลายในกรณีที่จะตรวจข้อผิดพลาดของโปรแกรม เพราะการที่ข้อมูลทั้งคำร้องขอและคำตอบรับถูกจับ"มัด" และส่งทีละก้อนใหญ่นั้น ทำให้เกิดกรณีประหลาดขึ้นได้หลายกรณี เช่น มีคำร้องถูกมัดติดกัน 3 คำร้องส่งไปยังผู้บริการและคำร้องที่หนึ่งเป็นตัวการทำให้เกิดการผิดพลาดในโปรแกรม ตัวผู้บริการจะไม่มีทางทราบได้ว่าคำร้องใดเป็นตัวก่อปัญหาจนกว่าคำร้องทั้งสามจะมาถึง(พร้อม ๆ กัน) หลังจากที่ค้นพบข้อผิดพลาดในคำร้องที่หนึ่งและผู้บริการทำการแจ้งเหตุไปยังโปรแกรมที่ส่งคำร้องมาแล้วก็จะต้องจัดการกับคำร้องที่สองและที่สามต่อไป กว่าที่รายงานความผิดพลาดจะกลับมาถึงโปรแกรมของเรา ผู้บริการก็ทำคำร้องอื่น ๆ เลยไปแล้วอีกมากมายทำให้การหาสาเหตุความผิดพลาดนั้นทำได้ไม่กระจ่างหรืออาจถึงขั้นสับสนสำหรับผู้ไม่มีความเข้าใจเรื่องระบบสื่อสารภายในของ Xlib ดังรูปที่ 2

## โปรแกรมตัวอย่าง

โปรแกรมตัวอย่างนี้ชื่อว่า hellow.c เป็นโปรแกรมที่ทำงานใกล้เคียงกับโปรแกรมแรกที่เราท่านเหล่าเขียนโปรแกรมภาษาซีรู้จักกันดี คือ hello.c โดยที่




**รูปที่ 2 ลักษณะการจัดคิวคำร้องและคำตอบรับในระบบ X-Window**

โปรแกรมซีมาตรฐานโปรแกรมนั้นจะพิมพ์ข้อความ Hello, World. ออกมา ส่วนโปรแกรมของเราก็จะออกข้อความเช่นเดียวกันเพียงแต่จะปรากฏขึ้นมาพร้อมกับหน้าต่างหนึ่งบาน แถมใช้เมาส์ได้อีกต่างหาก รวมทั้งยังรับรู้ว่าเรากดแป้นพิมพ์เป็นตัวอักษรใดได้ด้วย

การจะเขียนโปรแกรมใช้งานกับเอกซ์วินโดว์นั้น ต้องประกอบด้วยโปรแกรมชิ้นย่อยหลายส่วนและโปรแกรมย่อย ๆ เหล่านี้จะต้องมีความสัมพันธ์กันเป็นอย่างดีจึงจะส่งผลให้โปรแกรมโดยรวมทำงานได้ราบรื่น ขั้นตอนหลักอย่างต่ำที่สุดที่โปรแกรมหนึ่ง ๆ จะต้องทำเพื่อให้การทำงานเป็นไปอย่างปกติมีดังต่อไปนี้

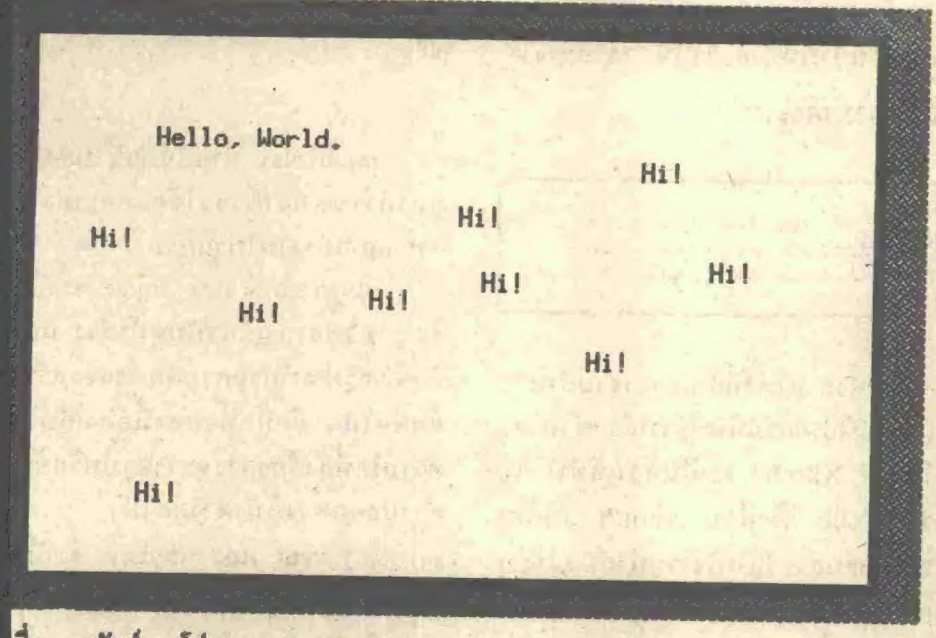
- ติดต่อกับผู้บริการ (X-server) ประจำเครื่อง
- สร้างหน้าต่างขึ้นมาหนึ่งบาน
- ติดต่อกับโปรแกรมจัดการหน้าต่าง (window manager) เพื่อแจ้งคุณสมบัติของหน้าต่างใหม่ที่สร้างขึ้นมา
- ติดต่อขอสร้างและจองทรัพยากรต่าง ๆ ที่ผู้บริการเป็นผู้รักษาไว้
- จัดการนำหน้าต่างที่สร้างขึ้นแล้วนั้นมาปรากฏบนจอ
- กำหนดขอบเขตของเหตุต่าง ๆ ที่แจ้งเข้ามา เลือกเฉพาะที่โปรแกรมของเราสนใจ
- จัดการอ่านเหตุการณ์ต่าง ๆ เหล่านั้น(ซึ่งอยู่ในแถวคิว)เข้ามา แล้วดำเนินการตามที่เราเห็นสมควร
- สร้างรูปภาพ(หรือตัวอักษร, ข้อความ ฯลฯ) ที่เราต้องการออกไปยังหน้าต่าง
- เมื่อสิ้นสุดโปรแกรมจะต้องจัดการ "ทำความสะอาด" โดยการคืนทรัพยากรต่าง ๆ แก่ผู้บริการ รวมถึงการปิดหน้าต่าง ฯลฯ

## การทำงานของ hellow.c

ดังที่กล่าวมาแล้วโปรแกรม hellow.c นี้ เมื่อใช้งานจะปรากฏหน้าต่างขึ้นมาหนึ่งบานที่มีข้อความว่า "Hello, World." อยู่ในนั้น ซึ่งถ้าผู้ใช้งานเลื่อนเมาส์เข้าไปในกรอบของหน้าต่างและกดปุ่มบนเมาส์ (ไม่ว่าปุ่มใด) จะปรากฏข้อความ "hi" ขึ้นตรงตำแหน่งที่เมาส์อยู่เมื่อเรากดปุ่ม ดังรูปที่ 3

ถ้ามีกรณีที่หน้าต่างของเราถูกหน้าต่างอื่นของโปรแกรมอื่นทับแล้วหลังจากนั้นก็ปล่อยให้หน้าต่างของเรา "ผุด" (expose) ขึ้นมาอีกครั้งหนึ่ง โปรแกรมของเราจะสามารถกู้ข้อความเริ่มต้นโปรแกรม (ซึ่งก็คือ "Hello, World.") กลับมาแสดงให้ดูได้ดังเดิม ในกรณีที่เราเลื่อนหรือขยับหน้าต่างไปก็จะปรากฏผลเช่นเดียวกัน ทั้งสองกรณีโปรดสังเกตว่าข้อความ "Hi" ทั้งหลายจะถูกลบไป ความสามารถอันนี้เป็นสิ่งสำคัญในโปรแกรมที่ใช้งานในระบบหน้าต่างหลายหน้าต่าง(ราชบัณฑิตเรียกพหุบัญชร) เนื่องจากหน้าต่างอื่น ๆ ของโปรแกรมอื่น ๆ หรือแม้แต่ของโปรแกรมเดียวกันนั้นจะมีเหตุให้หน้าต่างทั้งหลายกระทบกระทั่งกันอยู่ตลอดเวลา ไม่ว่าจะเป็น การทับ การซ้อน การบังเฉพาะส่วนใดส่วนหนึ่งหรืออื่น ๆ อีกมาก ดังนั้นโปรแกรมแต่ละโปรแกรมมีหน้าที่ดูแลสิ่งที่ปรากฏบนหน้าต่างของตนเอง และต้องสามารถแสดงออกมาใหม่ได้ทุกครั้งที่จำเป็น

นอกจากที่กล่าวมาแล้ว โปรแกรม hellow.c นี้ยังสามารถรับรู้การป้อนข้อมูลเข้าทางแป้นพิมพ์ได้อีกด้วย ซึ่งในโปรแกรมตัวอย่างนี้จะไม่สนใจอักขระอื่น ๆ นอกจากตัวอักษร "q" ซึ่งจะเป็นสัญญาณให้โปรแกรมจบการทำงาน จะเห็นได้ว่าโปรแกรมนั้นทำงานง่าย ๆ เพียงงานเดียวคือ แสดงข้อความ ทั้งโปรแกรมก็ไม่ใหญ่ประมาณร้อยบรรทัดเศษ แม้กระนั้นก็ยังเป็นโปรแกรมที่เขียนครบตามสูตรสำเร็จที่ได้กล่าวมาข้างต้นแล้ว




ที่ 3 ผลลัพธ์ของโปรแกรม hellow.c บนจอภาพ

## อธิบายโปรแกรม

ขั้นแรกเราจะมาดูโปรแกรมโดยคร่าว ๆ ก่อน โดยการแบ่งโปรแกรมเป็นส่วน ๆ แล้วหลังจากนั้นจะได้มีการอธิบายโปรแกรมกันในแต่ละส่วนต่อไป ซึ่งแรกสุดนี้เรายังไม่จำเป็นต้องเข้าใจอะไรทั้งหลายให้ทะลุโปร่งเสียทีเดียว เพียงแต่จะต้องการตัดแบ่งโปรแกรมออกเป็นส่วน ๆ ให้สมเหตุสมผลก่อน เพื่อให้เกิดจินตนาการกว้าง ๆ ซึ่งถ้าพิจารณาโปรแกรมตัวอย่างแล้วจะพอมองออกได้ว่า สามารถจะแบ่งเป็นส่วน ๆ ได้ดังนี้

- ส่วนท่อนหัวมาตรฐานของ X
- ส่วนจับจองตัวแปร ซึ่งรวมทั้งในส่วนกลาง (global) และในโปรแกรมหลัก
- ส่วนตั้งต้นโปรแกรม ซึ่งมีการติดต่อกับผู้บริการ สร้างหน้าต่าง จองทรัพยากรที่จะใช้ในการวาดรูปหรือพิมพ์ข้อความต่อไป
- ส่วนจำกัดขอบเขตการรับข้อมูลและการแจ้งเหตุ อันเป็นตัวบอกผู้บริการว่าโปรแกรมจะสนใจดูเหตุการณ์อะไรบ้าง
- การนำหน้าต่างไปปรากฏบนจอภาพ
- ส่วน วง(โค)จรหลัก หรือ main loop ซึ่งเป็นการอ่านเหตุการณ์ต่าง ๆ เข้ามา และแสดงผลทางจอภาพตามสถานการณ์ที่อ่านเข้ามาได้ ซึ่งประกอบด้วย

● การวาดหน้าต่างใหม่ทั้งหมดเมื่อมีเหตุการณ์ "ผุด" ขึ้นของหน้าต่างของเรา

● รับทราบการเปลี่ยนความหมายของแป้นพิมพ์

● อ่านและดำเนินการเมื่อมีการกดปุ่มเมาส์

● อ่านและดำเนินการเมื่อมีการกดแป้นพิมพ์

● ส่วนที่ต้องกระทำเมื่อสิ้นสุดโปรแกรม เช่น การคืนทรัพยากร การปิดการติดต่อกับผู้บริการ และการทำลายหน้าต่าง

จะสังเกตเห็นว่า ส่วนวงจรหลักนั่นแหละเป็นตัวจักรสำคัญที่กระทำทุกอย่างที่เราเห็นได้รู้สึกได้จากการใช้งานโปรแกรม ส่วนอื่นที่เหลือเป็นพวกจัดหลังฉากทั้งสิ้น เช่นส่วนตั้งต้นก็เป็นพวกที่จัดเวทีไว้ให้ ส่วนสุดท้ายก็เป็นพวกที่รื้อเวที ส่วนวงจรหลักนี้จะทำงานไปตามเหตุการณ์ที่มากระทบโปรแกรมเป็นหลัก เช่น มีการกดเมาส์เกิดขึ้น เป็นต้น ลักษณะการทำงานเช่นนี้คือยึดถือเหตุการณ์เป็นหลัก (event-directed) ต่างจากโปรแกรมทั่ว ๆ ไปที่จะตัดสินใจทำงานโดยยึดถือข้อมูลเป็นหลัก (data-directed) ซึ่งไม่เหมาะกับการใช้งานตอบโต้กับผู้ใช้

คราวนี้มาดูรายละเอียดของแต่ละส่วนกันว่ามันทำอะไรกันบ้าง อย่างไร

### ส่วนหัวมาตรฐาน

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xos.h>
```

ทั้งสามไฟล์นี้เป็นมาตรฐานสำหรับแทบทุกโปรแกรมที่จะใช้งานเอกซ์วินโดว์ โดยที่ Xlib.h จะเป็นส่วนที่ทำให้ใช้งาน Xlib ได้ ส่วน Xutil.h นั้นเป็นส่วนเสริมสำหรับให้ใช้งานฟังก์ชันต่าง ๆ เพิ่มเติมขึ้นกว่าปกติ ส่วน Xos.h นั้นเป็นส่วนเผื่อเหลือเผื่อขาดเนื่องจากระบบจัดการแต่ละตัวย่อมมีข้อแตกต่างกันอยู่บ้างเล็ก ๆ น้อย ๆ

### ส่วนจองที่ตัวแปร

```
char    hello[] = "Hello, World.
char    hi[]    = "Hi!";
```

สองตัวแปรนี้เป็นตัวแปรกลางเก็บข้อความซึ่งเป็นข้อความที่เราจะใช้ส่งออกจอ ดังรูปที่ 4

```
main(argc,argv)
int         argc;
char        **argv;
{
    Display         *mydisplay;
    Window          mywindow;
    GC              mygc;
    XEvent          myevent;
    KeySym          mykey;
    XSizeHints      myhint;

    int                 myscreen;
    unsigned long       myforegrou
    unsigned long       mybackgrou
    int                 i;
    char                text[10];
    int                 done;
```

รูปที่ 4

mydisplay นั้นเป็นตัวชี้ไปยังตัวผู้บริการของเราซึ่งจะใช้ตลอดเวลาในการติดต่อทำงานโปรแกรม

mywindow และ mygc จะเป็นข้อมูลทรัพยากรเกี่ยวกับหน้าต่าง และลักษณะรูปซึ่งโปรแกรมของเราจะสร้างขึ้นต่อไป ข้อมูลทั้งสองนี้ต้องเก็บไว้ตราบใดที่ยังต้องการจะวาดรูปหรือข้อความออกทางหน้าต่างต่อไป

myevent และ mykey จะเป็นตัวเก็บเหตุการณ์และตัวอักษรที่กดมาจากแป้นพิมพ์ตามลำดับ myhint นั้น

เป็นข้อมูลเกี่ยวกับขนาด กว้าง-ยาว และอื่น ๆ เพื่อให้โปรแกรมจัดการหน้าต่างและโปรแกรมอื่น ๆ รู้ถึงสภาพของหน้าต่างของเราขณะนั้น

myscreen เป็นตัวเก็บเลขที่ของจอภาพ (บางครั้งผู้บริการหนึ่งตัวคุมจอภาพมากกว่าหนึ่งจอ) myforeground และ mybackground นั้นเก็บสี ตัวแปรที่เหลือใช้เก็บข้อมูลเล็ก ๆ น้อย ๆ สำหรับใช้งานในโปรแกรม

## ส่วนตั้งต้นโปรแกรม

คำสั่ง XOpenDisplay เป็นการเริ่มต้นขอติดต่อกับผู้บริการ เป็นคำร้องขออันแรกของโปรแกรมซึ่งจะเป็นการขอเชื่อมโปรแกรมเข้ากับจอภาพ เมาส์ และแป้นพิมพ์ชุดหนึ่ง และเก็บข้อมูลไว้ใช้งานต่อไป ดังรูปที่ 5

คำสั่ง Defaultscreen จะเป็นการขอให้มีการจัดจอภาพให้หนึ่งจอ อะไรก็ได้ ส่วน WhitePixel และ BlackPixel จะถามว่าบนจอที่ขอมาได้นั้น สีขาวมีข้อมูลว่าอย่างไร สีดำมีข้อมูลว่าอย่างไร

ทั้งหมดนี้เป็นการขอกำหนดตำแหน่งของขอบซ้ายบนของหน้าต่าง และความกว้าง-สูงของหน้าต่างตามลำดับ สำหรับคำสั่งสุดท้ายเป็นการระบุว่าขนาดและตำแหน่งนั้นโปรแกรมเป็นผู้ใส่เข้าไป ซึ่งถ้าไม่มีคำสั่งนี้จะมีการหาข้อมูลกันเหนียวใส่เข้ามาไว้ให้ ดังรูปที่ 6

คำสั่งนี้เป็นการสร้างหน้าต่างขึ้นมาหนึ่งหน้าต่าง ซึ่งถือเป็นทรัพยากรอันหนึ่งเก็บไว้ที่ผู้บริการอ้างถึงได้ โดยตัวแปร mywindow เป็นหน้าต่างแบบธรรมดา มีขนาดตามกำหนด มีขอบกว้าง 5 จุดภาพและมีสีตามกำหนด ดังรูปที่ 7

เป็นคำสั่งอธิบายคุณสมบัติของหน้าต่างเพื่อให้โปรแกรมจัดการหน้าต่างและโปรแกรมข้างเคียงอื่น ๆ รับทราบ ดังรูปที่ 8

XCreateGC เป็นการขอสร้างทรัพยากรอีกหนึ่งตัว ซึ่งเก็บคุณสมบัติต่าง ๆ ของการวาดรูปหรือการเขียนข้อความฝากไว้กับผู้บริการ อ้างถึงโดยใช้ mygc จากนั้นก็เป็นการระบุว่าการวาดนั้นจะมีสีต่าง ๆ ตามที่กำหนด ดังรูปที่ 9

รูปที่ 5

```
mydisplay     = XOpenDisplay("");
myscreen      = DefaultScreen(mydisplay);
mybackground = WhitePixel(mydisplay, myscreen);
myforeground = BlackPixel(mydisplay, myscreen);
```

รูปที่ 6

```
myhint.x      = 200;
myhint.y      = 300;
myhint.width  = 350;
myhint.height = 250;
myhint.flags  = PPosition | PSize;
```

```
mywindow = XCreateSimpleWindow( mydisplay,
                                DefaultRootWindow(mydisplay),
                                myhint.x,
                                myhint.y,
                                myhint.width,
                                myhint.height,
                                5,
                                myforeground,
                                mybackground
                              );
```

รูปที่ 7

รูปที่ 8

```
XSetStandardProperties( mydisplay,
                        mywindow,
                        hello,
                        hello,
                        None,
                        argv,
                        argc,
                        &myhint);
```

```
mygc = XCreateGC(mydisplay, mywindow, 0, 0);
XSetBackground(mydisplay, mygc, mybackground);
XSetForeground(mydisplay, mygc, myforeground);
```

รูปที่ 9

```
XSelectInput( mydisplay,
              mywindow,
              ButtonPressMask | KeyPressMask | ExposureMask
            );
```

รูปที่ 10

**ส่วนจำกัดการรับรู้ข้อมูล**

เป็นการจำกัดว่าสำหรับหน้าต่างนี้เราจะรับรู้เหตุการณ์เฉพาะการกดปุ่มเมาส์ การกดแป้นพิมพ์ และการ "ผุด" ขึ้นเองหน้าต่างนี้เท่านั้น ดังรูปที่ 10

**ส่วนนำหน้าต่างไปปรากฏ**

เป็นการระบุว่า mywindow นี้มีข้อมูลครบถ้วนแล้ว สามารถนำไปสร้างเป็นหน้าต่างจริงบนจอภาพได้แล้ว หน้าต่างใหม่นี้จะ "ผุด" ขึ้นมาบนสุดทับหน้าต่างอื่น ๆ (ถ้ามี) และเป็นการแจ้งเหตุการณ์"ผุด"ในตัว ฉะนั้นเมื่อหน้าต่างโผล่ขึ้นมาบนจอภาพก็จะเริ่มทำงานตรงส่วนของโปรแกรมที่จัดการกับการ "ผุด" ไปด้วยในตัวเสร็จ ดังรูปที่ 11

**ส่วนวงจรหลัก**

จะมีการอ่านเหตุการณ์เข้ามาด้วยคำสั่ง XNextEvent ซึ่งจะได้เหตุการณ์มาเก็บไว้ใน myevent หลังจากนั้นจะมีการจัดการเหตุการณ์ตามชนิดของเหตุการณ์ต่อไป จะเป็นเช่นนี้ไปจนกว่า done จะไม่เท่ากับ 0 ซึ่งเราจะดูต่อไปว่าเกิดขึ้นตรงไหน ดังรูปที่ 12

การอ่านเหตุการณ์เข้ามาจะมีทางเลือกปฏิบัติได้หลายทางขึ้นอยู่กับตัวเหตุการณ์ดังนี้

**. กรณีมีการผุดขึ้นของหน้าต่าง**

การแจ้งเหตุ การ "ผุด" นี้บางครั้งมีมาซ้อน ๆ กันมากกว่าหนึ่งครั้ง ดังนั้นเราจะรอจนการแจ้งเหตุครั้งสุดท้ายมาถึงจึงจะ เขียนของความ "Hello, World." ใหม่บนหน้าต่างของเราด้วยคุณสมบัติตาม mygc ที่ตำแหน่ง (50, 50) ดังรูปที่ 13

**. กรณีมีการเปลี่ยนแป้นพิมพ์**

จะมีการอ่านแป้นพิมพ์ใหม่เข้ามาเพื่อให้มีการตีความแป้นพิมพ์ถูกต้อง ดังรูปที่ 14

**. กรณีมีการกดเมาส์**

จะมีการเขียนข้อความ "Hi!" ณ ตำแหน่งที่เมาส์ขึ้นอยู่ (ในหน้าต่าง) ด้วยคุณสมบัติตาม mygc จะเห็นได้ว่าข้อความ "Hello, World." และ "Hi!" นั้นเกิดจากเหตุคนละอันกัน ฉะนั้นเมื่อจอภาพถูกทับแล้วผุดขึ้นมาใหม่ บรรดา "Hi!" ทั้งหลายจะหายไปหมด ดังรูปที่ 15

รูปที่ 11

```
XMapRaised(mydisplay,mywindow);
```

```
done = 0;
while (done == 0)
{
    XNextEvent(mydisplay, &myevent);
    switch (myevent.type)
    {
```

รูปที่ 12

```
case Expose:    if (myevent.xexpose.count == 0)
                    XDrawImageString( myevent.xexpose.display,
                                      myevent.xexpose.window,
                                      mygc,
                                      50,
                                      50,
                                      hello,
                                      strlen(hello)
                                    );
                break;
```

รูปที่ 13

```
case MappingNotify:     XRefreshKeyboardMapping(&myevent);
                        break;
```

รูปที่ 14

```
case ButtonPress:    XDrawImageString( myevent.xbutton.display,
                                       myevent.xbutton.window,
                                       mygc,
                                       myevent.xbutton.x,
                                       myevent.xbutton.y,
                                       hi,
                                       strlen(hi)
                                     );
                     break;
```

รูปที่ 15

```
case KeyPress:      i = XLookupString( &myevent, text, 10, &mykey,
                    if (i == 1 && text[0] == 'q')
                       done = 1;
                    break;
```

รูปที่ 16

**. กรณีกดแป้นพิมพ์**

XLookupString จะอ่านการกดแป้นพิมพ์เข้ามาและแปลผลตาม mykey หลังจากนั้นจะเก็บไว้ในเท็กซ์โดยเก็บจำนวนตัวที่กดไว้ใน i ดังรูปที่ 16

หลังจากอ่านแป้นพิมพ์แล้วจะมาตรวจว่าตัวอักษรที่พิมพ์เข้ามาใช่ตัว "q" หรือไม่ ถ้าใช่เราจะให้ done เป็น 1 ซึ่งมีผลทำให้วงจรหลักหยุดทำงาน

**ส่วนปิดท้ายโปรแกรม**

คำสั่ง XFreeGC และ XDestroyWindow เป็นการคืนทรัพยากรให้กับผู้บริการสำหรับนำไปใช้กับโปรแกรมอื่น ๆ XCloseDisplay เป็นคำขอสุดท้ายคือตัดการติดต่อกับผู้บริการ นั่นคือเราจะไม่สามารถใช้งาน จอภาพ เมาส์ และแป้นพิมพ์ได้อีกต่อไป ส่วน exit(0); นั้นเพื่อให้มีการปิดไฟล์(เผื่อมีเราไม่ใช้) และทำความสะอาดอื่น ๆ รวมทั้งแจ้งให้ทราบ (สำหรับผู้อยากทราบ) ว่าโปรแกรมของเราเสร็จสมบูรณ์ไม่ผิดพลาด ดังรูปที่ 17

จะเห็นได้ว่าการจะเขียนโปรแกรมใช้งานในระบบเอกซ์วินโดว์นั้นไม่ยากแต่มาก! คือจะให้ออกข้อความสักสองอย่างเขียนกันร้อยกว่าบรรทัด แต่ผลลัพธ์ที่ได้สมใจอยากเหมือนกัน หวังว่าตัวอย่าง (รูปที่ 18) ที่ยกให้ดูคงพอจะทำให้เกิดความกระจ่างต่างหูต่างตากันบ้างเล็ก ๆ น้อย ๆ พอให้สงบจิตสงบใจรอคอยให้ระบบเอกซ์วินโดว์มาถึงมือคุณได้บ้าง หรืออาการจะกำเริบหนักกี่ไม่รู้สิ!

```
            } /* end switch */
        } /* end while */
        XFreeGC(mydisplay, mygc);
        XDestroyWindow(mydisplay, mywindow);
        XCloseDisplay(mydisplay);
        exit(0);
    } /* end main program */
```

รูปที่ 17

รูปที่ 18 hellow.c

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xos.h>


    char    hello[] = "Hello, World.";
    char    hi[]    = "Hi!";

main(argc,argv)
int         argc;
char        **argv;
{
    Display         *mydisplay;
    Window          mywindow;
    GC              mygc;
    XEvent          myevent;
    KeySym          mykey;
    XSizeHints      myhint;

    int                 myscreen;
    unsigned long       myforeground;
    unsigned long       mybackground;
    int                 i;
    char                text[10];
    int                 done;


    mydisplay    = XOpenDisplay("");
    myscreen     = DefaultScreen(mydisplay);
    mybackground = WhitePixel(mydisplay, myscreen);
    myforeground = BlackPixel(mydisplay, myscreen);

    myhint.x      = 200;
    myhint.y      = 300;
    myhint.width  = 350;
```

รูปที่ 18 (ต่อ)

```
        myhint.height = 250;
        myhint.flags  = PPosition | PSize;

        mywindow = XCreateSimpleWindow( mydisplay,
                                        DefaultRootWindow(mydisplay),
                                        myhint.x,
                                        myhint.y,
                                        myhint.width,
                                        myhint.height,
                                        5,
                                        myforeground,
                                        mybackground
                                      );
       XSetStandardProperties( mydisplay,
                               mywindow,
                               hello,
                               hello,
                               None,
                               argv,
                               argc,
                               &myhint);
       mygc = XCreateGC(mydisplay, mywindow, 0, 0);
       XSetBackground(mydisplay, mygc, mybackground);
       XSetForeground(mydisplay, mygc, myforeground);
       XSelectInput( mydisplay,
                     mywindow,
                     ButtonPressMask | KeyPressMask | ExposureMask
                   );
       XMapRaised(mydisplay,mywindow);


       done = 0;
       while (done == 0)
       {
           XNextEvent(mydisplay, &myevent);
           switch (myevent.type)
           {
               case Expose:    if (myevent.xexpose.count == 0)
                                  XDrawImageString( myevent.xexpose.display,
                                                    myevent.xexpose.window,
                                                    mygc,
                                                    50,
                                                    50,
                                                    hello,
                                                    strlen(hello)
                                                  );
                               break;
               case MappingNotify:    XRefreshKeyboardMapping(&myevent);
                                      break;
               case ButtonPress:    XDrawImageString( myevent.xbutton.display,
                                                      myevent.xbutton.window,
                                                      mygc,
                                                      myevent.xbutton.x,
                                                      myevent.xbutton.y,
                                                      hi,
                                                      strlen(hi)
                                                    );
                                    break;
               case KeyPress:     i = XLookupString( &myevent, text, 10, &mykey, 0);
                                  if (i == 1 && text[0] == 'q')
                                     done = 1;
                                  break;
           } /* end switch */
       } /* end while */
       XFreeGC(mydisplay, mygc);
       XDestroyWindow(mydisplay, mywindow);
       XCloseDisplay(mydisplay);
       exit(0);
   } /* end main program */
```